Non-isomorphic spanning trees of graphs

Janneke van den Boomen

July 2009

Radboud University Nijmegen

Non-isomorphic spanning trees of graphs

Janneke van den Boomen

Master Thesis
Student number: 0314064
Supervisor: Dr. W. Bosma
Second Reader: Dr. R.H. Jeurissen
Faculty of Science
Radboud University Nijmegen

Contents

Chapter 1

Introduction

In a connected graph G, it is (usually) easy to find a tree that contains all the vertices and some edges of G; such a subgraph is called a spanning tree. With the *Matrix tree theorem* [10], we can determine how many spanning trees a graph contains. Looking at the proof of that theorem, it even seems not very difficult[1] to find all those spanning trees (which also can be done in other ways). But what can we say about the isomorphism classes of the spanning trees? How many and which non-isomorphic spanning trees does the graph contain? In this thesis, we will look at those questions.

In chapter 2, we will look at trees and graphs from different points of view, trying to discover properties which can tell us something about two graphs being isomorphic or not. This since we want to partition the set of spanning trees into isomorphism classes.

Chapter 3 is about the algorithm that determines all the non-isomorphic spanning trees of a given graph.

Now, we can determine all the non-isomorphic spanning trees of a graph. But maybe there exists a formula which tells us how many there are. For an arbitrary graph, it is hard to determine such a formula. However, for bipartite graphs it seems doable. In chapter 4, we take a look at earlier results and use them to determine a formula for the number of non-isomorphic spanning trees of $K_{4,t}$.

[1]Theoretically... For big graphs it still takes a while.

Finally, in chapter 5, we take a look at two extreme cases. First, we look at graphs that contain all possible non-isomorphic spanning trees. What is the minimum size of such graphs? We will discuss a way for finding them and take a look at some results.
Second, there is the set of graphs of which all spanning trees are isomorphic. What do they look like? How many edges do they contain? In section 5.2, we will see which graphs have just one non-isomorphic spanning tree and how many of those graphs there are.

In the entire thesis, we only talk about simple, finite and connected graphs. (A graph is simple if it contains just single edges and no loops.)

Chapter 2

Isomorphisms

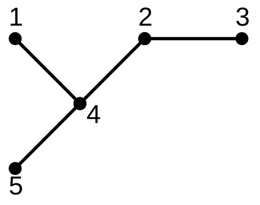
To see which non-isomorphic spanning trees a graph contains, we need to know when two trees are isomorphic. Therefore, we will look at trees and graphs from different points of view, trying to discover properties that can tell us something about two graphs being isomorphic or not.

2.1 Trees

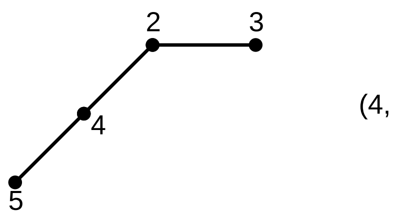
2.1.1 Prüfer sequence

A *Prüfer sequence* of a tree with n labeled vertices is a sequence of length $n-2$ on the labels 1 to n. A Prüfer sequence is generated as follows. At step i, remove the leaf (vertex with degree 1) with the smallest label and set the i^{th} element of the sequence to be the label of this leaf's neighbour.

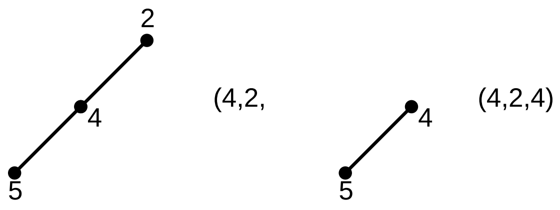
Example 2.1 **(a)** *We take the tree below.*



The leaf with the smallest label is 1*. Its neighbour is* 4*, so this is the first element of the Prüfer sequence. Then we delete this leaf from the tree.*



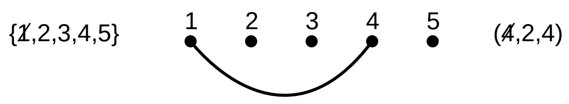
We repeat these steps until there are just 2 *vertices left:*



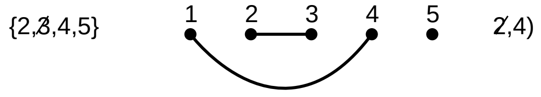
So our tree can be associated with the sequence $(4, 2, 4)$.

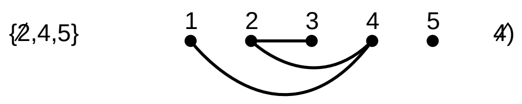
With this Prüfer sequence, we can construct a tree. Let V be the remaining set of vertices (initially: $\{1, 2, \ldots, n\}$) and PS the remaining Prüfer sequence (initially a sequence of length $n - 2$ on the labels 1 to n). At every step, we take the first number of PS and connect this vertex with the smalles number in V that does not appear in PS. Then we delete those two numbers from PS and V respectively.
If PS is empty, we will have two vertices left in V. Connecting them is the last step of the algorithm.

Example 2.1 (b) *We start with the Prüfer sequence* $(4, 2, 4)$.
Since the sequence is of length 3, *we have* 5 *vertices. So we start with vertexset* $V = \{1, 2, \ldots, 5\}$. *The first number in the Prüfer sequence is* 4 *and the smallest number in* V *that does not appear in the Prüfer sequence is* 1. *So we get:*

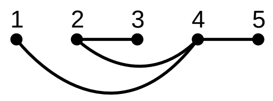


We repeat this step until our sequence is empty.





Finally, we connect the numbers left in V:



We see that in example 2.1(b), we get the same tree as the tree we started with in example 2.1(a). So to every labeled tree T belongs a unique Prüfer sequence PS_T. With this we can see that there is a bijection between sequences of length $n-2$ on the labels 1 to n, and labeled trees on n vertices [11].

Can we say something about two trees being isomorphic or not, using these Prüfer sequences?
Since the numbers that occur in a Prüfer sequence depend on how we label the tree, we can't look at *which* numbers occur. But maybe we can look at *how many times* a number occurs.

Definition 2.2 *Let PS_{T_1} and PS_{T_2} be two Prüfer sequences of length $n-2$. We say $PS_{T_1} \approx PS_{T_2}$ if for every j with $0 \leq j \leq n-2$, the number of labels that occur j times is the same for both sequences.*

Example 2.3 *Let $n = 6$, $PS_{T_1} = (1, 2, 5, 2)$ and $PS_{T_2} = (3, 6, 1, 1)$. Then $PS_{T_1} \approx PS_{T_2}$ since:*
3 numbers occur 0 times (in the first sequence, 3, 4 and 6 are missing; in the second sequence 2, 4 and 5 are missing),
2 numbers occur 1 time (in the first sequence 1 and 5; in the second sequence 3 and 6),
1 number occur 2 times (in the first sequence 2; in the second sequence 1),
0 numbers occur 3 times,
0 numbers occur 4 times.

Hypothesis 2.4 *Let T_1 and T_2 be two labeled trees on n vertices and PS_{T_1} and PS_{T_2} their corresponding Prüfer sequences. Then*

$$T_1 \cong T_2 \Longleftrightarrow PS_{T_1} \approx PS_{T_2}.$$

$\Rightarrow$ is true. This we can see with the next theorem.

Theorem 2.5 *Let T be a tree. Then for every vertex v of T holds*

$$\deg_T(v) = k \Longleftrightarrow v \textit{ appears } k-1 \textit{ times in } PS_T$$

Proof:
We will prove this with induction on n, the number of vertices of T.

- $n = 2$: then T is just an edge, so $PS_T = ()$.
  For vertex 1 we have: $\deg_T(1) = 1$ and 1 appears zero times in PS_T.
  For vertex 2 the same holds.
  So for $n = 2$, the theorem holds.

- Suppose the theorem holds for n vertices.
  Let T be a tree with $n + 1$ vertices.
  Let v_1 be the leaf with the smallest label, v_2 its neighbour and v_i some random other vertex.
  Let T' be the tree that results from deleting v_1 from T.

  The Prüfer sequence of T starts with v_2, and then follows the Prüfer sequence of T'.

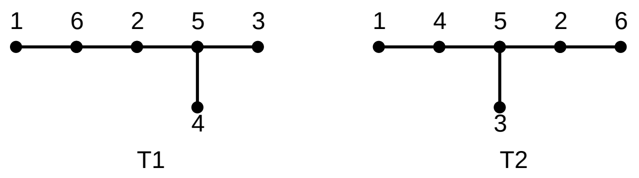
  - $\deg_T(v_1) = 1$ and v_1 appears zero times in $PS_{T'}$ (because v_1 is not a vertex of T'), so it appears zero times in PS_T.
  - $\deg_T(v_2) = \deg_{T'}(v_2) + 1$ and (because of induction) v_2 appears $\deg_{T'}(v_2)-1$ times in $PS_{T'}$, so it appears $\deg_{T'}(v_2) = \deg_T(v_2)-1$ times in PS_T.
  - Because of induction, v_i appears $\deg_{T'}(v_i) - 1$ times in $PS_{T'}$, so also $\deg_{T'}(v_i) - 1$ times in PS_T. And $\deg_{T'}(v_i) = \deg_T(v_i)$.

  So for a tree with $n + 1$ vertices, the theorem holds.

So, by induction, the theorem holds for every tree. $\square$

Since the degree sequences of two isomorphic trees T_1 and T_2 are the same, it is trivial that then $PS_{T_1} \approx PS_{T_2}$.

Unfortunataly, $\Longleftarrow$ in 2.4 is not true. Take the two trees below.



For these trees, we have

$$PS_{T_1} = (6,5,5,2) \approx (4,5,5,2) = PS_{T_2},$$

but the trees are not isomorphic.
This is the smallest counterexample for 2.4 (with respect to the number of vertices).

2.1.2 Determinant of the reduced incidence matrix

For every graph G, we can define its incidence matrix.

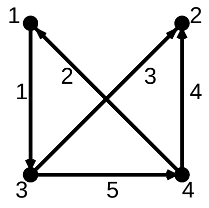
Definition 2.6 *Let G be a graph with n vertices and m edges (numbered arbitrarily). We orient each edge randomly. The* incidence matrix *of G is the $n \times m$ matrix $A_G = [a_{ij}]$ with*

$$a_{ij} = \begin{cases} +1 & \textit{if the } j^{th} \textit{ edge is oriented to the } i^{th} \textit{ vertex} \\ -1 & \textit{if the } j^{th} \textit{ edge is oriented away from the } i^{th} \textit{ vertex} \\ 0 & \textit{otherwise} \end{cases}$$

If you number the vertices or edges in an other way, the rows or columns of the matrix A_G will be permuted. If you orient an edge otherwise, the $+1$ and -1 will be swapped. So the matrix A_G will not really be different.

From this matrix, we can make a *reduced incidence matrix* $\widetilde{A_G}$ of G. This matrix we make by deleting the nth row from the matrix A_G.

Example 2.7 *Take the graph G below (in which we have numbered the vertices already, and oriented and numbered the edges).*



The matrices of this graph are:

$$A_G = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 \end{pmatrix} \quad \textit{and} \quad \widetilde{A_G} = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 & -1 \end{pmatrix}$$

The reduced incidence matrix of a tree is an $(n-1) \times (n-1)$-matrix.

Hypothesis 2.8 *Let T_1 and T_2 be two trees, both with n vertices.*

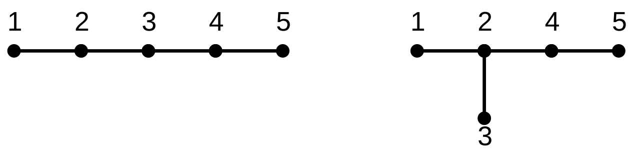
$$T_1 \cong T_2 \Longleftrightarrow \det(\widetilde{A_{T_1}}) = \det(\widetilde{A_{T_2}})$$

Since the labeling of the vertices and edges is arbitrary, the hypothesis does certainly not hold. Take, for example, a tree T and his incidence matrix A_T. Let T' be the same tree as T, but with the labels of the first two vertices swapped. Then we get $A_{T'}$ from A_T by interchanging the first and second row. So, it is obvious that T and T' are isomorphic, but $\det(\widetilde{A_T}) = -\det(\widetilde{A_{T'}})$. So we change our statement into the next one.

Hypothesis 2.9 *Let T_1 and T_2 be two trees, both with n vertices.*

$$T_1 \cong T_2 \Longleftrightarrow |\det(\widetilde{A_{T_1}})| = |\det(\widetilde{A_{T_2}})|$$

Now, $\Rightarrow$ is true. However, $\Leftarrow$ is not true. We can see this with the two trees below.



T1 T2

The reduced incendence matrices are

$$\widetilde{A_{T_1}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix} \quad \text{and} \quad \widetilde{A_{T_2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

So $|\det(\widetilde{A_{T_1}})| = |1| = 1 = |1| = |\det(\widetilde{A_{T_2}})|$, but the trees are not isomorphic.

From the Matrix Tree Theorem [10], we can deduce that for every tree T holds $|\det(\widetilde{A_G})| = 1$.

Theorem 2.10 (Matrix Tree Theorem) *If $\widetilde{A_G}$ is a reduced incidence matrix of the connected graph G, then the number of spanning trees of G equals the determinant of $\widetilde{A_G} \cdot \widetilde{A_G}^T$.*

Since a tree only has one spanning tree, we have

$$\begin{aligned} 1 &= \det(\widetilde{A_G} \cdot \widetilde{A_G}^T) \\ &= \det(\widetilde{A_G}) \cdot \det(\widetilde{A_G}^T) \\ &= \det(\widetilde{A_G})^2 \end{aligned}$$

So $|\det(\widetilde{A_{T_1}})| = |\det(\widetilde{A_{T_2}})|$ holds for every pair of trees.

2.1.3 Label of the center

The *center* of a graph is the set of vertices that minimize the maximal distance from all other vertices. For a tree, the center is quite simple [5]:

Theorem 2.11 *In a tree, the center is a vertex or two adjecent vertices (i.e. an edge).*

Proof:
We will prove this with induction on n, the number of vertices of the tree.

- $n \leq 2$: if the tree has at most two vertices, the center is the entire tree.

- Suppose the theorem holds for all trees with less than n vertices. Let T be a tree with n vertices. We get T' from T by deleting all leaves. Every vertex at maximum distance in T from a vertex u is a leaf. Let $\epsilon_T(u)$ be this maximum distance in T. Since all the leaves have been removed and no path between two other vertices uses a leaf, $\epsilon_{T'}(u) = \epsilon_T(u) - 1$ for every vertex u of T'. Also, $\epsilon_T(u) > \epsilon_T(v)$ if u is a leaf and v its neighbour. Hence, the vertices minimizing $\epsilon_T(u)$ are the same as the vertices minimizing $\epsilon_{T'}(u)$. So T and T' have the same center. And since T' has fewer vertices than T, the center of T' is a vertex or an edge. Hence, the center of T is a vertex or an edge.

So, by induction, the theorem holds for every tree. $\square$

This proof also gives us an easy algorithm to find the center of a tree T:

```
Repeat, until we have one or two vertices:
    Remove all leaves from $T$
```

Since the size of the center of a tree is 1 or 2, there is not much variation. Hence, it's hard to say something about two trees being isomorphic based on the size of their centers. The only thing we can say is that if two trees are isomorphic, then their centers have to be of the same size.

But instead of just determining the center, we also can *label* the center, and compare those labels. How we label the trees is based on an algorithm to determine isomorphism between two trees [8]. The labeling algorithm goes as follows:

Algorithm LabelCenter

1. *Label all the leaves with* 1.

2. *Determine the set S of unlabeled vertices that have at most one unlabeled neighbour. Tentatively, label the vertices in S with the list $[l_1, \ldots, l_k]$ of labels of its labeled neighbours, sorted in non-decreasing order.*

3. *Substitute the tentative labels (which are lists of numbers) by new labels which are just numbers: order the tentative labels in non-decreasing order. The vertices with the smallest label get the label m, where m is the smallest number that has not been used as a label before. The*

vertices with then the smallest label get the label $m+1$. In this way, all vertices with the same tentative labels get the same new label and all new labels have not been used before.

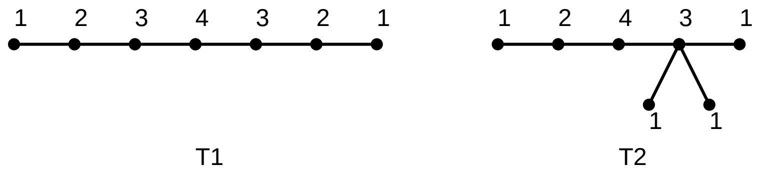
4. *If not all vertices have got labels, go back to step 2.*

In this way, every vertex gets a label, so also the vertices in the center (these vertices are the last labeled). The label of the center of the tree T, LC_T, is the ordered sequence of the labels of the vertices in the center.

Hypothesis 2.12 *Let T_1 and T_2 be two trees.*

$$T_1 \cong T_2 \iff LC_{T_1} = LC_{T_2}$$

It is easy to see that $\Rightarrow$ is true. But $\Leftarrow$ is not true: take the trees below.



The numbers at the vertices are the labels the vertices get in the algorithm discribed above. So despite T_1 and T_2 are obviously not isomorphic,

$$LC_{T_1} = [4] = LC_{T_2}.$$

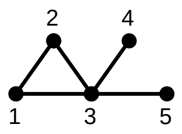
This is the smallest counterexample for the hypothesis.

Remark: the algorithm from [8] (algorithm TreeIsomorphism *in section 3.2) on which the label of the center is based of course tells us whether two trees are isomorphic or not. But if we want to divide a given set of trees into isomorphism classes, just determine LC_T of every tree and compare them is much faster than run the algorithm for* every *pair of trees.*

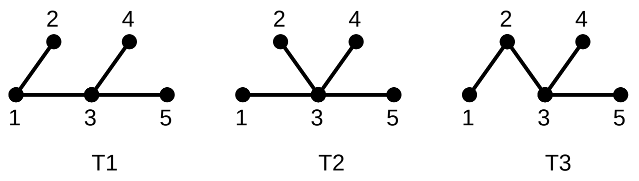
2.2 Graphs

2.2.1 Tree sequence

In this section, we will explain everything with the same example: the graph G below.

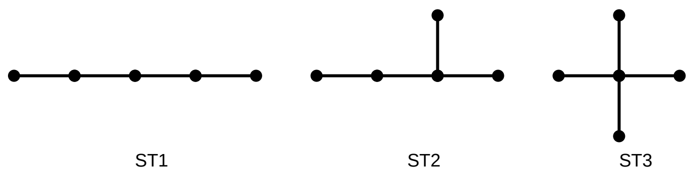


Every graph contains some spanning trees. How many can be determined with the *Matrix Tree Theorem* (Theorem 2.10). In our example occur the next spanning trees:



Some of the spanning trees are isomorphic, so we can put them in isomorphismclasses. For our graph G, we have two isomorphismclasses: $\{T1, T3\}$ and $\{T2\}$.

The next thing we have to do is determine all the non-isomorphic trees on n vertices. For $n = 5$, we have the trees



Then we fix their order (for $n = 5$, the order is as above).

Now we can define the tree sequence of a graph.

Definition 2.13 *Let H be a graph with n vertices. Let A be the sequence of all non-isomorphic spanning trees on n vertices, of which we have fixed the order. The* tree sequence *of this graph, TS_H, is a sequence of length $\#A$ with*

$$TS_H[i] := \textit{number of spanning trees in } H\textit{, isomorphic to } A[i]$$

In our example, we have 0 times the tree $ST1$, 2 times $ST2$ and 1 time $ST3$, so

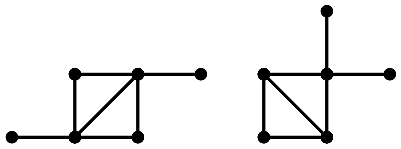
$$TS_G = [0, 2, 1].$$

Hypothesis 2.14 *Let G_1 and G_2 be two graphs. Then*

$$G_1 \cong G_2 \Longleftrightarrow TS_{G_1} = TS_{G_2}$$

If two graphs are isomorphic, it is obvious that they contain the same spanning trees and with the same frequency. So $\Rightarrow$ holds.

$\Leftarrow$ does not hold. A smallest counterexample (smallest order ánd smallest size) is



The corresponding tree sequences are:

$$TS_{G1} = [0, 2, 0, 0, 4, 2] = TS_{G2}.$$

Remark: To compare the sizes of the isomorphism classes of the spanning trees of two graphs, it is not necessary to determine all the non-isomorphic trees on n vertices. You also can detemine the isomorphism classes of the spanning trees of the first graph and the sizes of those classes, and then look whether the second graph contains the same non-ismorphic spanning trees with the same multiplicity.

2.2.2 Ordered incidence matrix

As we have seen in section 2.1.2, to every graph G belongs an incidence matrix A_G. Actually, to every graph belong many incidence matrices: just label the edges or vertices in a different way or orient the edges different and you get an other matrix.
Can we order the rows and columns, such that we get a unique matrix?
We define the next order:

Let A_G be an incidence matrix of G.

1. Order the rows by degree (the degree is the number of non-zero entries in a row).

2. Order the columns lexicographic (where a non-zero entry is bigger than a zero entry, so $[1,-1,0,0] > [1,0,-1,0] > [0,1,-1,0]$).
3. Order every set of rows of the same degree lexicographic (the same way as defined in step 2).

   Now we get the ordered incidence matrix A^o_G.

To give a better idea of how this ordering works, we will give an example.

Example 2.15 *Take the incidence matrix*

$$A_G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \end{pmatrix}$$

Ordering this matrix step by step, we get:

$$A_G = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \end{pmatrix} \xrightarrow{(1)} \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}$$

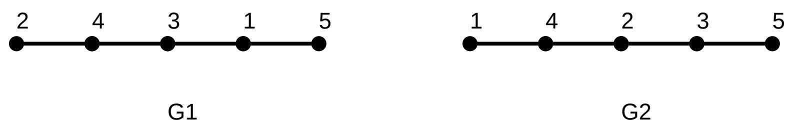
$$\xrightarrow{(2)} \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \end{pmatrix} \xrightarrow{(3)} \begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = A^o_G.$$

We wonder if the ordered incidence matrix is unique, i.e. if the next hypothesis holds.

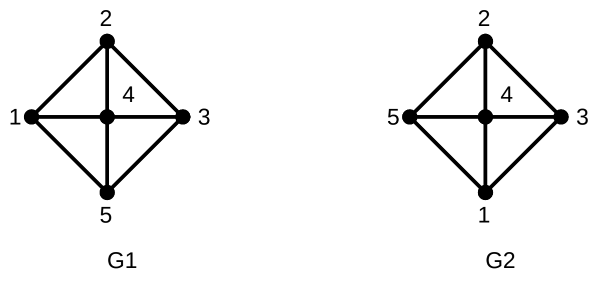
Hypothesis 2.16 *Let G_1 and G_2 be two graphs. Then*

$$G_1 \cong G_2 \iff A^o_{G_1} = A^o_{G_2}$$

Since the matrix tells us exactly which vertices are adjacent, $\Leftarrow$ is trivial. But $\Rightarrow$ does not hold. The smallest counterexample is shown below.



Another counterexample is



The ordered incendence matrices are

$$A^o_{G1} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & \mathbf{-1} & 0 & 1 & \mathbf{0} \\ 0 & 0 & -1 & 0 & \mathbf{0} & 0 & -1 & \mathbf{1} \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 \end{pmatrix}$$

and

$$A^o_{G2} = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & \mathbf{0} & 0 & 1 & \mathbf{1} \\ 0 & 0 & -1 & 0 & \mathbf{-1} & 0 & -1 & \mathbf{0} \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & -1 \end{pmatrix}$$

It is clear that $G1$ and $G2$ are isomorphic. But $A^o_{G1} \neq A^o_{G2}$ (they differ in the entries which are in bold).

Chapter 3

Determine all non-isomorphic spanning trees

To determine all non-isomorphic spanning trees of a graph, we will use the algorithm NIST. This algorithm works about the same as how we made the tree sequences in section 2.2.1.

```
Algorithm NIST
    Input: labeled graph G
        1. Determine all spanning trees of G
        2. Partition all spanning trees into isomorphism classes
    Output: all not-isomophic spanning trees of G
```

Both steps in the algorithm will be explained in the next sections. The Magma-code of all the algorithms can be found in [13].

3.1 Determine all spanning trees

D.E. Knuth [7] describes an algorithm, called *Algorithm S*, to determine all spanning trees of a graph.
Knuth uses a special notation of a graph: he replaces all edges by two contrary directed edges. Then he translates the picture-notation into a notation with sequences of succesors, predecessor, etc. In this section, we will get an idea of how the algorithm works. To do that, we will not use Knuths notation, but the original notation (the picture of the undirected graph).

The main thing used in the algorithm is Lemma 3.2. Before we describe the theorem, we need a definition.

Definition 3.1 *Let G be a graph and $e = uv$ an edge of G. G/e is the graph we get by shrinking e to one vertex, i.e. we delete the edge(s) between u and v and identify u and v.*
Graphs we get from shrinking an edge are allowed to have multiple edges.

If e is part of a particular spanning tree, then the other $n - 2$ edges span the graph G/e. So we have the next theorem.

Lemma 3.2 *Let G be a graph and e an edge of G.*
The spanning trees of G that contain e are the same as the spanning trees of G/e to which we add e.

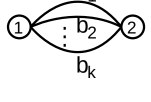
We start the algorithm with a (random) spanning tree, say $a_1, \ldots, a_{n-1}$.
All spanning trees can be classified in the next $n - 1$ classes:

1. Trees with $a_2, \ldots, a_{n-1}$.
2. Trees with $a_2, \ldots, a_{n-2}$ and without a_{n-1}.
3. Trees with $a_2, \ldots, a_{n-3}$ and without a_{n-2}.

$\vdots$

$n-2$. Trees with a_2 and without a_3.
$n-1$. Trees without a_2.

We will start with looking for all the spanning trees in the first class. Therefore, we shrink the edges $a_2, \ldots, a_{n-1}$. What is left is a graph with two vertices and between them k edges $b_1, \ldots, b_k$, like below.



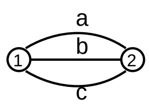
The remaining edges all are a spanning tree of the new graph, so (by Lemma 3.2) together with the edges $a_2, \ldots, a_{n-1}$ they are a spanning tree of G. So we found the spanning trees $a_2 \cdots a_{n-1} b_1$, $a_2 \cdots a_{n-1} b_2$, ..., $a_2 \cdots a_{n-1} b_k$.

Now we will look at the second class: trees with $a_2, \ldots, a_{n-2}$ and without a_{n-1}.
Because we want spanning trees without a_{n-1}, we first delete this edge. In the first class, we started with a near-tree, but this time we only have fixed $n-3$ edges. Therefore, we first look for all trees with $a_2, \ldots, a_{n-2}, b_k$ (b_k because that edge was extra in the last spanning tree we found). This goes similar as in the first class, and we find the trees $a_2 \cdots a_{n-2} b_k c_1$, $a_2 \cdots a_{n-2} b_k c_2$, ..., $a_2 \cdots a_{n-2} b_k c_m$.
But we were looking for all spanning trees with $a_2, \ldots, a_{n-2}$, not with $a_2, \ldots,$ a_{n-2}, b_k. So the next step is to find all spanning trees with $a_2, \ldots, a_{n-2}$ and without b_k and a_{n-1}. So we delete also edge b_k and start looking for all trees with $a_2, \ldots, a_{n-2}, c_m$ (again, c_m because this is the extra edge in the last spanning tree we found).
At some point, there are no trees without a_{n-1} and all the extra edges we used (for example b_k and c_m). Then we have all trees in the second class.
To test whether or not there are trees without some set of edges, every time we delete an edge we check whether this edge is a bridge or not.

Definition 3.3 *Let G be a graph and e an edge of G. Then e is a* bridge *if G gets disconnected when we delete e.*

Finding the spanning trees in the third and fourth class goes the same.

In the algorithm, Knuth also uses l_{edge}'s and s_{level}'s. We replace those numbers by sequences L and S.

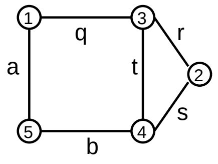
> In the sequence L, the places are represented by the edges (such that $L[edge] = l_{edge}$). So if the order of the edges in L is $e_1, e_2, \ldots, e_k$, then $L[e_2]$ is on the second place in L.
> With this sequence, we record which edges also *have to be* shrunk of we shrink a specific edge.
> For example, we have to shrink edge a in the graph below.
>
> 
>
>
> Then we also have to shrink b and c, which we have to record because we have to know this when we unshrink a. We record this as follows: $L[a] := b, L[b] := c, L[c] := 0$. So, when we have to unshrink a, we do that and then look at $L[a]$. If this is zero, we are done. If this is not

zero, put $L[a]$ also back in the graph. Then we check whether $L[L[a]]$ is zero or not, etc.

The sequence S is such that $S[level] = s_{level}$. With this sequence, we record the last deleted edge in the level we are in. The edges deleted before are recorded in L, in the same way as the forced shrinked edges are recorded.

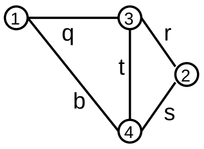
To explain the algorithm a bit more, we will give an example.

Example 3.4 *In this example, we will use the graph below.*

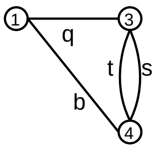


The order of the edges in L is q, a, r, s, t, b.

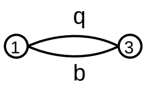
We start the algorithm with a (random) spanning tree: qart.
1. All trees with art
First, we shrink the near tree (a set of $n-2$ edges and no cycle).



shrink a
$S = [0]$
$L = [, 0, , , ,]$



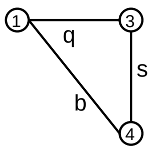
shrink r
$S = [0, 0]$
$L = [, 0, 0, , ,]$



shrink t
$S = [0, 0, 0]$
$L = [, 0, 0, 0, s,]$

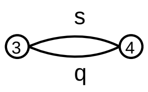
The remaining edges are spanning trees of the remaining graphs, so with art they are a spanning tree of the whole graph.
Trees are artq and artb.

Unshrink t (and because $L(t) = s$, also unshrink s ($L(s) = 0$)).



t is not a bridge, so delete t.
$S = [0, 0, t]$
$L = [, 0, 0, 0, 0,]$

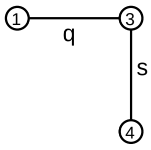
2. All trees with ar and without t
Start with all trees with arb (we take b as extra edge, because that was in the last tree we found) and not t.



shrink b
$S = [0, 0, t]$
$L = [, 0, 0, 0, 0, 0]$

Trees are $arbq$ and $arbs$.

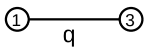
Now unshrink b ($L(b) = 0$).



b is not a bridge, so delete b.
$S = [0, 0, b]$
$L = [, 0, 0, 0, 0, t]$

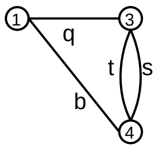
All trees with ars and not t or b.

shrink s
$S = [0, 0, b]$
$L = [, 0, 0, 0, 0, t]$



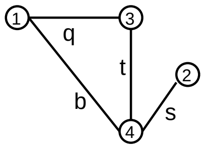
Trees are $arsq$.

Unshrink s ($L(s) = 0$).



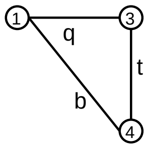
s is a bridge, so undelete $S[3] = b$ (because we are in level 3 now) and $L(b) = t$ ($L(t) = 0$).
$S = [0, 0, b]$
$L = [, 0, 0, 0, 0, t]$

(Enter level 2)
Unshrink r ($L(r) = 0$)

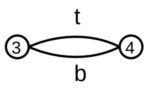


r is not a bridge, so delete r.
$S = [0, r, b]$
$L = [, 0, 0, 0, 0, t]$

3. All trees with a and without r
Start with all trees with asq and not r.



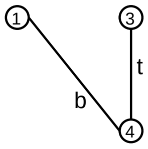
Shrink s
$S = [0, r, b]$
$L = [, 0, 0, 0, 0, t]$



Shrink q.
$S = [0, r, 0]$
$L = [0, 0, 0, 0, 0, t]$

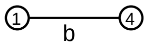
All trees are $asqb$ and $asqt$.

Unshrink q $(L(q) = 0)$.



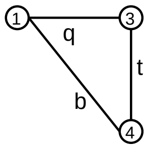
q is not a bridge, so delete q.
$S = [0, r, q]$
$L = [0, 0, 0, 0, 0, t]$

Now determine all trees with ast and not r and q.
Shrink t.
$S = [0, r, q]$
$L = [0, 0, 0, 0, 0, t]$



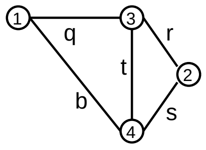
All trees are $astb$.

Unshrink t $(L(t) = 0)$.



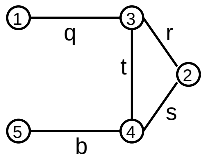
t is a bridge, so undelete $S(3) = q$ $(L(q) = 0)$.
$S = [0, r, q]$
$L = [0, 0, 0, 0, 0, t]$

Unshrink s $(L(s) = 0)$.



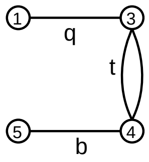
s is a bridge, so undelete $S(e) = r$ ($L(r) = 0$).
$S = [0, r, q]$
$L = [0, 0, 0, 0, 0, t]$

(Enter level 1)
Unshrink a.

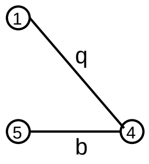


a is not a bridge, so delete a.
$S = [a, r, q]$
$L = [0, 0, 0, 0, 0, t]$

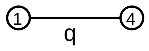
4. All trees without a
Start with all trees with stb and not a.



Shrink s.
$S = [a, r, q]$
$L = [0, 0, 0, 0, 0, t]$

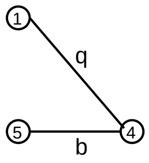


Shrink t.
$S = [a, 0, q]$
$L = [0, 0, 0, 0, r, t]$



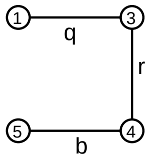
Shrink b.
$S = [a, 0, 0]$
$L = [0, 0, 0, 0, r, 0]$

Unshrink b ($L(b) = 0$).



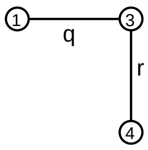
b is a bridge, so undelete (there are no edges to undelete).
$S = [a, 0, 0]$
$L = [0, 0, 0, 0, r, 0]$

Unshrink t ($L(t) = r$ so also unshrink r ($L(r) = 0$)).



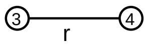
t is not a bridge, so delete t.
$S = [a, t, 0]$
$L = [0, 0, 0, 0, 0, 0]$

All trees with sbq and not a and t.



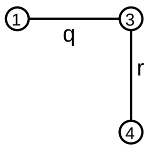
Shrink b.
$S = [a, t, 0]$
$L = [0, 0, 0, 0, 0, 0]$

Shrink q.
$S = [a, t, 0]$
$L = [0, 0, 0, 0, 0, 0]$



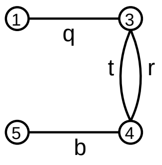
All trees are sbqr.

Unshrink q (L(q) = 0).



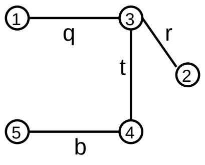
q is a bridge, so undelete (there are no edges to undelete).
$S = [a, t, 0]$
$L = [0, 0, 0, 0, 0, 0]$

Unshrink b (L(b) = 0).



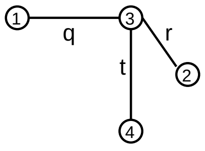
b is a bridge, so undelete t (L(t) = 0).
$S = [a, t, 0]$
$L = [0, 0, 0, 0, 0, 0]$

Unshrink s (L(s) = 0).

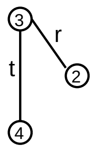


s is not a bridge, so delete s.
$S = [s, t, 0]$
$L = [0, 0, 0, a, 0, 0]$

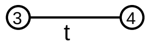
All trees with bqr and not a and s.



Shrink b.
$S = [s, t, 0]$
$L = [0, 0, 0, a, 0, 0]$



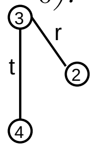
Shrink q.
$S = [s, 0, 0]$
$L = [0, 0, 0, a, 0, 0]$



Shrink r.
$S = [s, 0, 0]$
$L = [0, 0, 0, a, 0, 0]$

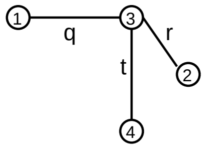
All trees are $bqrt$.

Unshrink r ($L(r) = 0$).



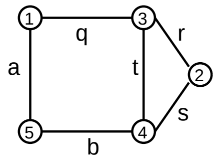
r is a bridge, so undelete (there are no edges to undelete).
$S = [s, 0, 0]$
$L = [0, 0, 0, a, 0, 0]$

Unshrink q ($L(q) = 0$).



q is a bridge, so undelete (there are no edges to undelete).
$S = [s, 0, 0]$
$L = [0, 0, 0, a, 0, 0]$

Unshrink b ($L(b) = 0$).



b is a bridge, so undelete s ($L(s) = a$ so also undelete a ($L(a) = 0$)).
$S = [s, 0, 0]$
$L = [0, 0, 0, a, 0, 0]$

So all trees are:

$artq$	$asqb$
$artb$	$asqt$
	$astb$
$arbq$	
$arbs$	$stbq$
$arsq$	$sbqr$
	$bqrt$

3.2 Partition into isomorphism classes

Now, with *Algorithm S*, we can construct the set of all spanning trees of our graph G. The next step is to divide them into ismorphism classes, and pick one tree from every class. This we will do with the algorithm TreeClasses.

Algorithm TreeClasses

```
Input: AllTrees, the sequence of all spanning trees of graph G
    Classes := [AllTrees[1]];
    for i := 2 to #AllTrees do
        Classes := NewClasses(AllTrees[i], Classes);
    end for;
Output: Classes, the sequence of all non-isomorphic spanning trees of G.
```

While running the algorithm, *Classes* contains one tree of every isomorphism class of the first $i-1$ trees of *AllTrees*. The algorithm NewClasses checks whether the i^{th} tree is isomorphic to one of the trees in *Classes* or not. In the first case, the new sequence *Classes* is the same as the old set. In the second case, we found a tree non-isomorphic to the earlier trees, so we have a new isomorphism class. So then the new sequence *Classes* is the old one with *AllTrees*[i] included.

To check whether two trees T_1 and T_2 are isomorphic or not, we use the next algorithm from [8].

Algorithm TreeIsomorphism

1. *In each graph, label all the leaves with* 1. *If the numbers of leaves in* T_1 *and* T_2 *do not coincide, stop with "non-isomorphic".*
2. *In each graph, determine the sets of unlabeled vertices* S_1 *and* S_2 *such that every neighbour of* v *in* S_i*, except at most one, has a label. Tentatively, label* v *with the list* $[l_1, \ldots, l_k]$ *of labels of its labeled neighbours, sort in non-decreasing order. Compare the respective tentative labels for the vertices in* S_1 *and* S_2*. If these labels do not agree (as multisets), stop with "non-isomorphic".*
3. *In each graph, substitute the tentative labels (which are lists of numbers) by new labels which are just numbers: order the tentative labels of both graphs in non-decreasing order. The vertices with the smallest label get the new label* m*, where* m *is the smallest number that has not been used*

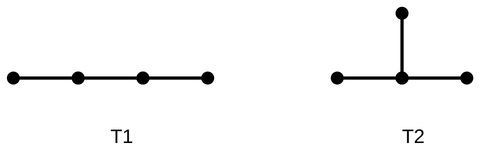
as a label before. The vertices with then the smallest label get the new label $m+1$. In this way, all vertices with the same tentative labels get the same new label and all new labels have not been used before.

4. *If not all vertices have got labels, go back to step 2.*

5. *Stop with "isomorphic".*

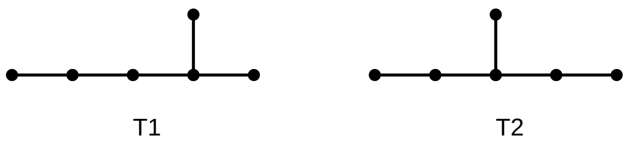
To see how the algorithm works, we will look at some examples. In the first two examples, we will see that the given trees are not isomorphic.

Example 3.5

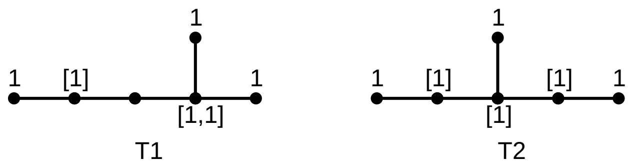


Since the numbers of leaves of the trees above are different, the algorithms will return "non-isomorphic" in the first step.

Example 3.6



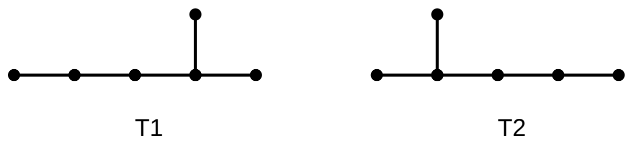
After step 2 (the first time), the labels will be like below.



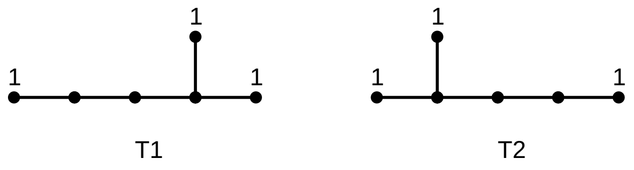
The tentative labels of T_1 are $\{[1],[1,1]\}$ and the tentative labels of T_2 are $\{[1],[1],[1]\}$. Since these labels are not the same, the algorithm will return "non-isomorphic" in step 2.

The last example shows how the algorithm works on two isomorphic trees.

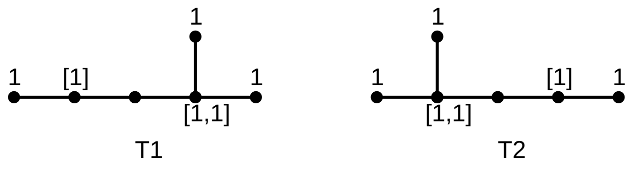
Example 3.7



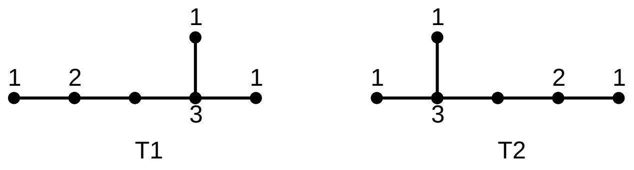
$\downarrow$ *Step* 1



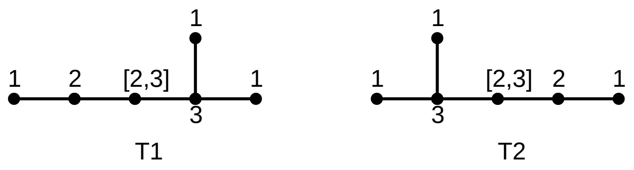
$\downarrow$ *Step* 2



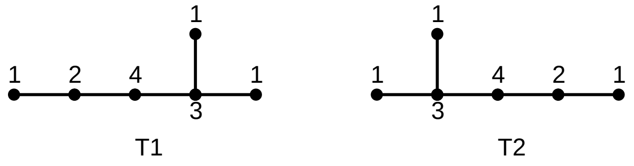
$\downarrow$ *Step* 3



Since not all vertices are labeled, we return to step 2.



$\downarrow$ *Step* 3



And since all the labels are the same in every step, the trees are isomorphic.

Chapter 4

Bipartite graphs

For some classes of graphs, such as trees and cycles, it is obvious what the number of non-isomorphic spanning trees is. For an arbitrary graph, it is hard to determine a formula for the number of non-isomorphic spanning trees. However, for complete bipartite graphs it turned out te be doable.

Definition 4.1 *Let G be a graph. Then $I(G)$ is the number of non-isomorphic spanning trees of G.*

In [9], Mohr proved the formulas for $I(K_{2,t})$ and $I(K_{3,t})$ he found.
In the first section of this chapter, we will introduce all the definitions and theorems we need to prove those formulas. We also will show what the formulas are. In the second section, we will prove the formula for $I(K_{4,t})$.

4.1 Earlier results

Selecting the possible spanning trees, we often have to split the set of t vertices (make a partition). To count the number of possibilities to do that, we will use *placing unlabeled balls (vertices) into (un)labeled boxes.* We will use the following definition and propositions (from [9] and [2]).

Definition 4.2 *$p_k(n)$ is the number of partitions of n into at most k parts.*

This is the same the number of ways to divide n unlabeled balls into k unlabeled boxes.

Proposition 4.3 *The number of ways to arrange n balls into k boxes is showed in the table below.*

balls	*boxes*	*number of empty boxes*	*number of ways*
unlabeled	*unlabeled*	≥ 0	$p_k(n)$
unlabeled	*unlabeled*	0	$p_k(n) - p_{k-1}(n)$
unlabeled	*labeled*	≥ 0	$\binom{n+k-1}{n}$
unlabeled	*labeled*	0	$\binom{n-1}{k-1}$
unlabeled	*unlabeled*	$\leq k-2$	$p_k(n) - 1$
unlabeled	*labeled*	$\leq k-2$	$\binom{n+k-1}{n} - k$

(The last two cases are the situations where at least two boxes are nonempty.)

The generating function of $p_k(n)$ is $\frac{1}{(1-x)(1-x^2)\cdots(1-x^k)}$.

Proposition 4.4 *With the generating function, we can determine*

$$p_2(n) = \lfloor \tfrac{n}{2} \rfloor + 1$$

and

$$p_3(n) = \tfrac{1}{72}(6n^2 + 36n + 47 + 9(-1)^n + 8c)$$
$$\text{where } c = \begin{cases} 2 & \text{if } n \equiv 0 \mod 3 \\ -1 & \text{if } n \equiv 1,2 \mod 3 \end{cases}$$

Furthermore, we will use the next proposition (also from [9]).

Proposition 4.5

$$\sum_{k=2}^{n} \left\lfloor \frac{k}{2} \right\rfloor = \begin{cases} \frac{n^2}{4} & \text{if } n \text{ even} \\ \frac{n^2-1}{4} & \text{if } n \text{ odd} \end{cases}$$

As we have seen in Theorem 2.11, the center of every tree is one vertex or two adjecent vertices (i.e. an edge). So in $K_{s,t}$, the center is a vertex in the s-set, a vertex in the t-set or an edge between the s-set and t-set.
Mohr also proved a really useful proposition about these centers:

Proposition 4.6 *Let T be a tree and $P_r(v)$ be the length of the unique rv-path in T. The vertex r is the unique center of T if and only if there exist at least two vertices v_i such that:*

1. $P_r(v_i)$ is maximal among all the vertices of T

2. the rv_i-paths are disjoint except for the common vertex r.

With these propositions it is possible to prove the next results (we will not do the proof, but the proof of the theorem in the next section will go in the same way as the proof of these formulas).

Theorem 4.7 *Let $T \geq 1$. Then*

(i) $I(K_{1,t}) = 1$;

(ii) $I(K_{2,t}) = \lceil \frac{t}{2} \rceil$;

(iii) $I(K_{3,t}) = \frac{1}{9}(3t^2 + 3t + 1 + c)$ *where*

$$c = \begin{cases} 2 & \text{if } t \equiv 1 \pmod 3 \\ -1 & \text{if } t \equiv 0, 2 \pmod 3). \end{cases}$$

4.2 $K_{4,t}$

For $K_{4,t}$ we found a closed formula for the number of the isomorphism classes. The proof is similar to the proof of $I(K_{2,t})$ and $I(K_{3,t})$ in [9].

As we have seen before, the generating function of $p_k(n)$ is $\frac{1}{(1-x)(1-x^2)\cdots(1-x^k)}$. In [1] we can find

Proposition 4.8 *We can rewrite the function $\frac{1}{(1-x)(1-x^2)\cdots(1-x^k)}$ into*

$$\sum \frac{1}{1^{p_1}2^{p_2}\cdots k^{p_k} \cdot p_1!p_2! \cdot p_k!} \cdot \frac{1}{(1-x)^{p_1}(1-x^2)^{p_2} \cdot (1-x^k)^{p_k}}$$

where the summation runs over all partitions $1 \cdot p_1 + 2 \cdot p_2 + \cdots + k \cdot p_k$ of k.

With this formula we can determine $p_4(n)$.

Theorem 4.9

$$p_4(n) = \frac{1}{288}\Big(2n^3 + 30n^2 + 135n + 175 + (45 + 9n)(-1)^n + 36z_1 + 32z_2\Big)$$

where $z_1 = \begin{cases} (-1)^{\frac{n}{2}} & \text{if } n \text{ is even} \\ 0 & \text{if } n \text{ is odd.} \end{cases}$ *and* $z_2 = \begin{cases} 1 & \text{if } n \equiv 0 \mod 3 \\ 0 & \text{if } n \equiv 1 \mod 3 \\ -1 & \text{if } n \equiv 2 \mod 3 \end{cases}$

Proof:
With the formula of proposition 4.8 and partial fraction decomposition, we get

$$\begin{aligned}
p_4(n) &= [x^n]\frac{1}{(1-x)(1-x^2)(1-x^3)(1-x^4)} \\
&= [x^n]\Big(\frac{\frac{1}{24}}{(1-x)^4} + \frac{\frac{1}{4}}{(1-x)^2(1-x^2)} + \frac{\frac{1}{8}}{(1-x^2)^2} + \frac{\frac{1}{3}}{(1-x)(1-x^3)} + \frac{\frac{1}{4}}{(1-x^4)}\Big) \\
&= [x^n]\Big(\frac{1}{24}\cdot\frac{1}{(1-x)^4} + \frac{1}{8}\cdot\frac{1}{(1-x)^3} + \frac{59}{288}\cdot\frac{1}{(1-x)^2} + \frac{17}{72}\cdot\frac{1}{(1-x)} \\
&\qquad + \frac{1}{8}\cdot\frac{1}{(1+x)} + \frac{1}{32}\cdot\frac{1}{(1+x)^2} + \frac{1}{8}\cdot\frac{1}{1+x^2} + \frac{1}{9}\cdot\frac{1+x}{1+x+x^2}\Big)
\end{aligned}$$

Now, we have to determine the coefficient of the n^{th} term of each fraction. The cases $\frac{1}{(1-x)^i}$ are easy: $\frac{1}{(1-x)^i} = (\frac{1}{1-x})^i = (1+x+x^2+x^3+\cdots)^i$. So the n^{th} coefficient is the number of ways we we can make n out of i numbers (≥ 0), which is $\binom{n+i-1}{n} = \binom{n+i-1}{i-1}$.
We can rewrite the fraction $\frac{1}{1+x}$ into $\frac{1}{1-(-x)} = 1-x+x^2-x^3+\cdots$, so the n^{th} coefficient is $(-1)^n$.
With this formula, we can see that the n^{th} coefficient of $\frac{1}{(1+x)^2} = (\frac{1}{1+x})^2$ is $(n+1)(-1)^n$.
And, in the same way, the n^{th} coefficient of $\frac{1}{1+x^2}$ is $z_1 = \begin{cases} (-1)^{\frac{n}{2}} & \text{if } n \text{ is even} \\ 0 & \text{if } n \text{ is odd.} \end{cases}$
The last fraction is equal to $\frac{1+x}{1+x+x^2} = \frac{(1+x)(1-x)}{1-x^3} = \frac{(1-x^2)}{1-x^3} = (1-x^2)(1+x^3+x^6+\cdots) = 1-x^2+x^3-x^5+\cdots$. So the n^{th} coefficient is

$$z_2 = \begin{cases} 1 & \text{if } n \equiv 0 \mod 3 \\ 0 & \text{if } n \equiv 1 \mod 3 \\ -1 & \text{if } n \equiv 2 \mod 3 \end{cases}$$

So

$$\begin{aligned}
p_4(n) &= \frac{1}{24}\binom{n+3}{3} + \frac{1}{8}\binom{n+2}{2} + \frac{59}{288}\binom{n+1}{1} + \frac{17}{72} + \frac{1}{8}(-1)^n \\
&\qquad + \frac{1}{32}(n+1)(-1)^n + \frac{1}{8}z_1 + \frac{1}{9}z_2 \\
&= \frac{1}{288}\Big(2n^3 + 30n^2 + 135n + 175 + (45+9n)(-1)^n + 36z_1 + 32z_2\Big)
\end{aligned}$$

where $z_1 = \begin{cases} (-1)^{\frac{n}{2}} & \text{if } n \text{ is even} \\ 0 & \text{if } n \text{ is odd.} \end{cases}$ and $z_2 = \begin{cases} 1 & \text{if } n \equiv 0 \mod 3 \\ 0 & \text{if } n \equiv 1 \mod 3 \\ -1 & \text{if } n \equiv 2 \mod 3 \end{cases}$

$\square$

Theorem 4.10 *For $t \geq 5$,*

$$I(K_{4,t}) = \frac{29}{144}t^3 + \frac{13 + 5 \cdot (-1)^{t-1}}{32} \cdot t + f + g$$

where

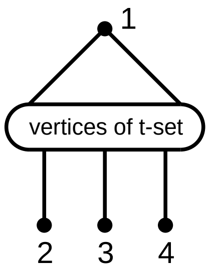
$$f = \begin{cases} 0 & \text{if } t \equiv 0 \mod 3 \\ \frac{1}{9} & \text{if } t \equiv 1 \mod 3 \\ -\frac{1}{9} & \text{if } t \equiv 2 \mod 3 \end{cases}$$

and

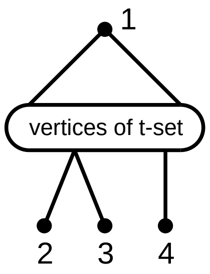
$$g = \begin{cases} 0 & \text{if } t \text{ is even} \\ \frac{1}{8}(-1)^{\frac{t-1}{2}} & \text{if } t \text{ is odd.} \end{cases}$$

Proof:
We will denote the 4-set by $\{1, 2, 3, 4\}$ and t-set by $\{v_1, v_2, \ldots, v_t\}$ (in fact, the vertices are not labeled, but this makes it easier to illustrate the proof).

Case: unique center in 4-set Let the vertex 1 be the center. The possible spanning trees are showed below.

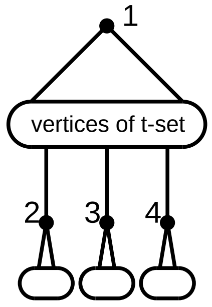


In this case, there is only one tree.



Also in this case, there is just one tree.

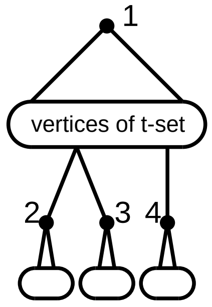
The case in which the vertices 2, 3 and 4 are adjecent to the same vertex of the t-set we will see later (*case: center is an edge*).



Now, we can put $n - k$ vertices in the upper subset of the t-set and (together) k vertices in the 3 subsets below. Then we have to divide those k vertices into 3 unlabeled boxes with at least 2 nonempty: $p_3(k) - 1$ possibilities.
Since at least two boxes have to be nonempty, k must be bigger or equal to 2. And since the upper box has to have at least 3 vertices, we have $k \leq t - 3$. So in this case we have

$$\sum_{k=2}^{t-3} (p_3(k) - 1) = \sum_{k=2}^{t-3} p_3(k) - t + 4$$

possible trees.

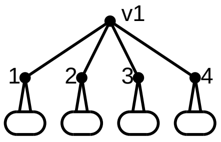


Here, we put $n-k$ vertices in the upper subset, l in de two subsets under 2 and 3 and $k - l$ in the subset under vertex 4. The only thing we have to do then, is divide the l vertices into two boxes. Since the upper subset and the subset under 4 have to be nonempty, we get

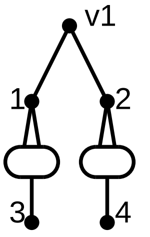
$$\sum_{k=2}^{t-2} \sum_{l=1}^{k-1} p_2(l)$$

trees.

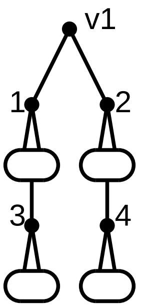
Case: unique center in t-set Now we take v_1 to be the center.



The number of spanning trees is of course $p_4(t - 1) - 1$ (the -1 at the end is because we must have at least 2 nonempty boxes).



Here we have to arrange $t-1$ vertices into 2 boxes, at least two (so both) nonempty. This gives us $p_2(t-1)-1 = \lfloor\frac{t-1}{2}\rfloor$ trees.



Put k vertices into the upper 2 unlabeled boxes and $t-1-k$ in the lowest 2 (also unlabeled). Because all boxes have to be nonempty, $2 \leq k \leq t-3$. If we divide the k vertices into the sets A_1 and A_2 and the $t-1-k$ into the sets B_1 and B_2, two things can happen. First, if $|A_1| = |A_2|$ or $|B_1| = |B_2|$, we get the tree $1-A_1-3-B_1$ with $2-A_2-4-B_2$. This can only happen if k respectively $t-1-k$ is even. Second, if $|A_1| \neq |A_2|$ and $|B_1| \neq |B_2|$, we get two different trees: $1-A_1-3-B_1$ with $2-A_2-4-B_2$ and $1-A_1-3-B_2$ with $2-A_2-4-B_1$. So then we have to multiply the result by 2.
We call $a(k)$ the number of possibilities to split k in two sets of equal size, and $b(k)$ the number of possibilities to split k in two non-empty sets of different size. If k is odd, $a(k) = 0$ and $b(k) = p_2(k) - 1$. If k is even, $a(k) = 1$ and $b(k) = p_2(k) - 1 - a(k) = p_2(k) - 2$.
So we have four cases:

1. *k even, $t-1-k$ odd*
   The number of possible trees is $2 \cdot b(k) \cdot b(t-1-k) + a(k) \cdot b(t-1-k) = 2(p_2(k)-2)(p_2(t-1-k)-1) + 1 \cdot (p_2(t-1-k)-1) = 2(\lfloor\frac{k}{2}\rfloor - 1)\lfloor\frac{t-1-k}{2}\rfloor + \lfloor\frac{t-1-k}{2}\rfloor = 2\lfloor\frac{k}{2}\rfloor\lfloor\frac{t-1-k}{2}\rfloor - \lfloor\frac{t-1-k}{2}\rfloor = \frac{(k-1)(t-k-2)}{2}$.

2. *k odd, $t-1-k$ even*
   Similar to "*k even, $t-1-k$ odd*", we get $2\lfloor\frac{k}{2}\rfloor\lfloor\frac{t-1-k}{2}\rfloor - \lfloor\frac{k}{2}\rfloor = \frac{(k-1)(t-k-2)}{2}$.

3. *k odd, $t-1-k$ odd*
   In this case, it cannot occur that we divide k or $t-1-k$ into two sets of the same size, so now we simply have $2(p_2(k)-1)(p_2(t-1-k)-1) = 2\lfloor\frac{k}{2}\rfloor\lfloor\frac{t-1-k}{2}\rfloor == \frac{(k-1)(t-k-2)}{2}$ possibilities.

4. *k even, $t-1-k$ even*
   The number of possible trees is $2\cdot b(k)\cdot b(t-1-k)+a(k)\cdot b(t-1-k)+b(k)\cdot a(t-1-k)+a(k)\cdot a(t-1-k) = 2(p_2(k)-2)(p_2(t-1-k)-2)+(p_2(k)-2)+(p_2(t-1-k)-2)+1 = 2\lfloor\frac{k}{2}\rfloor\lfloor\frac{t-1-k}{2}\rfloor-\lfloor\frac{k}{2}\rfloor-\lfloor\frac{t-1-k}{2}\rfloor+1 == \frac{(k-1)(t-k-2)}{2}-\frac{1}{2}$.

If t is even, only case 1 and 2 occur. So the number of possible trees is

$$\sum_{k=2}^{t-3}\frac{(k-1)(t-k-2)}{2} = \frac{1}{2}\sum_{k=2}^{t-3}\big(tk-k^2-k-t+2\big) = \frac{1}{12}\big(t^3-9t^2+26t-24\big)$$

If t is odd, only case 3 and 4 occur. So the number of possible trees is

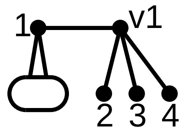
$$\sum_{k=2}^{t-3}\frac{(k-1)(t-k-2)}{2}+\sum_{\substack{2\le k\le t-3\\ k\text{ even}}}\frac{1}{2} = \frac{1}{12}\big(t^3-9t^2+26t-24\big)+\frac{1}{2}\frac{t-3}{2}$$

So, in general, the number of possible trees is

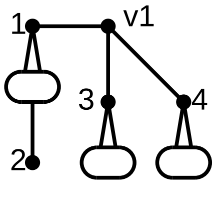
$$\frac{1}{12}\big(t^3-9t^2+26t-24\big)+h$$

where $h = \begin{cases} 0 & \text{if } t \text{ even}\\ \frac{t-3}{4} & \text{if } t \text{ odd}\end{cases}$

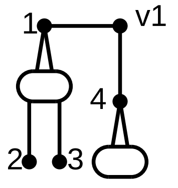
Case: center is an edge



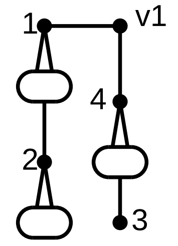
In this case, there is of course just one spanning tree.



We can put k vertices in the two unlabeled boxes on the right and $t-1-k$ in the left box. Since the left box have to be nonempty, and of the two right boxes at least one has to be nonempty, $1\le k\le t-2$. So the number of spanning trees is $\sum_{k=1}^{t-2}p_2(k)$.



We have to divide the $t-1$ vertices into two labeled boxes, both nonempty: $\binom{(t-1)-1}{2-1} = t-2$. Then there is just one case that cannot happen: when the left subset contains just one vertex. So there are $t-3$ possible trees. The vertices 2 and 3 can also be adjecent to the same vertex. Then but subsets have to be nonempty. So now the number of spanning trees is $t-2$.



Here we arrange the $t-1$ vertices into three labeled boxes, all nonempty. So there are $\binom{(t-1)-1}{3-1} = \binom{t-2}{2} = \frac{1}{2}(t-2)(t-3)$ spanning trees.

So, summing up the results above, we get

$$\begin{aligned}I(K_{4,t}) &= 1+1+\sum_{k=2}^{t-3} p_3(k) - t + 4 + \sum_{k=2}^{t-2}\sum_{l=1}^{k-1} p_2(l)\\
&\quad + p_4(t-1) - 1 + \left\lfloor \frac{t-1}{2} \right\rfloor + \frac{1}{12}\big(t^3 - 9t^2 + 26t - 24\big) + h\\
&\quad + 1 + \sum_{k=1}^{t-2} p_2(k) + 2t - 5 + \frac{1}{2}(t-1)(t-2)\\
&= t^2 - 2t + 3 + p_4(t-1) + \sum_{k=2}^{t-3} p_3(k) + \frac{1}{12}\big(t^3 - 9t^2 + 26t - 24\big) + h\\
&\quad + (t-1)\sum_{k=1}^{t-3}\left\lfloor \frac{k}{2}\right\rfloor - \sum_{k=1}^{t-3} k\left\lfloor\frac{k}{2}\right\rfloor\end{aligned}$$

where $h = \begin{cases} 0 & \text{if } t \text{ even} \\ \frac{t-3}{4} & \text{if } t \text{ odd}\end{cases}$

Since

(1)

$$\begin{aligned}
\sum_{k=2}^{t-3} p_3(k) &= \frac{1}{72}\Big(6\sum_{k=2}^{t-3} k^2 + 36\sum_{k=2}^{t-3} k + \sum_{k=2}^{t-3} 47 + 9\sum_{k=2}^{t-3}(-1)^k + 8\sum_{k=2}^{t-3} c\Big) \\
&= \frac{1}{72}\Big(6(\frac{1}{6}(t-3)(t-2)(2t-5)-1) \\
&\quad + 36\cdot\frac{1}{2}(t-3-2+1)(t-3+2) + 47(t-4) \\
&\quad + 9\sum_{k=2}^{t-3}(-1)^k + 8\sum_{k=2}^{t-3} c\Big) \\
&= \frac{1}{72}\Big(2t^3+3t^2-6t-152+a+b\Big)
\end{aligned}$$

where $a = \begin{cases} 0 & \text{if } t \text{ is even} \\ 9 & \text{if } t \text{ is odd.} \end{cases}$ and $b = \begin{cases} 8 & \text{if } t \equiv 0 \mod 3 \\ 0 & \text{if } t \equiv 1 \mod 3 \\ -8 & \text{if } t \equiv 2 \mod 3 \end{cases}$

(2) For odd t (and so $t-3$ is even), we have

$$\begin{aligned}
\sum_{k=1}^{t-3}\Big\lfloor\frac{k}{2}\Big\rfloor &= 2\sum_{i=1}^{\lfloor\frac{t-3}{2}\rfloor} i - \lfloor\frac{t-3}{2}\rfloor \\
&= \Big\lfloor\frac{t-3}{2}\Big\rfloor\Big(\Big\lfloor\frac{t-3}{2}\Big\rfloor+1\Big) - \Big\lfloor\frac{t-3}{2}\Big\rfloor \\
&= \frac{t-3}{2}\frac{t-3}{2} \\
&= \frac{1}{4}(t^2-6t+9)
\end{aligned}$$

For even t (and so $t-3$ is odd), we have

$$\begin{aligned}
\sum_{k=1}^{t-3}\Big\lfloor\frac{k}{2}\Big\rfloor &= 2\sum_{i=1}^{\lfloor\frac{t-3}{2}\rfloor} i \\
&= \Big\lfloor\frac{t-3}{2}\Big\rfloor\Big(\Big\lfloor\frac{t-3}{2}\Big\rfloor+1\Big) \\
&= \frac{t-4}{2}\frac{t-2}{2} \\
&= \frac{1}{4}(t^2-6t+8)
\end{aligned}$$

So in general

$$\sum_{k=1}^{t-3} \left\lfloor \frac{k}{2} \right\rfloor = \frac{1}{4}(t^2 - 6t + 9) + d$$

where $d = \begin{cases} -\frac{1}{4} & \text{if } t \text{ even} \\ 0 & \text{if } t \text{ odd} \end{cases}$

(3) If we want to rewrite $\sum_{k=1}^{t-3} k \left\lfloor \frac{k}{2} \right\rfloor$, we again have to look at t mod 2. If t is even (so $t-3$ is odd), we get:

$$\begin{aligned} \sum_{k=1}^{t-3} k \left\lfloor \frac{k}{2} \right\rfloor &= \sum_{i=1}^{\lfloor \frac{t-3}{2} \rfloor} i(2i + 2i + 1) \\ &= \sum_{i=1}^{\lfloor \frac{t-3}{2} \rfloor} (4i^2 + i) \\ &= \frac{1}{24}(4t^3 - 33t^2 + 86t - 72) \end{aligned}$$

If t is odd (so $t-3$ is even), we get:

$$\begin{aligned} \sum_{k=1}^{t-3} k \left\lfloor \frac{k}{2} \right\rfloor &= \sum_{i=1}^{\lfloor \frac{t-3}{2} \rfloor - 1} i(2i + 2i + 1) + (t-3) \left\lfloor \frac{t-3}{2} \right\rfloor \\ &= \sum_{i=1}^{\lfloor \frac{t-3}{2} \rfloor - 1} (4i^2 + i) + (t-3) \left\lfloor \frac{t-3}{2} \right\rfloor \\ &= \frac{1}{24}(4t^3 - 33t^2 + 92t - 87) \end{aligned}$$

So we can rewrite the formula as below.

$$\sum_{k=1}^{t-3} k \left\lfloor \frac{k}{2} \right\rfloor = \frac{1}{24}(4t^3 - 33t^2 + 86t - 72) + h$$

where $h = \begin{cases} 0 & \text{if } t \text{ even} \\ \frac{1}{4}t - \frac{5}{8} & \text{if } t \text{ odd} \end{cases}$

So we have

$$\begin{aligned}
I(K_{4,t}) &= t^2 - 2t + 3 + p_4(t-1) + \sum_{k=2}^{t-3} p_3(k) + \frac{1}{12}\big(t^3 - 9t^2 + 26t - 24\big) + h \\
&\quad + (t-1)\sum_{k=1}^{t-3}\left\lfloor \frac{k}{2} \right\rfloor - \sum_{k=1}^{t-3} k \left\lfloor \frac{k}{2} \right\rfloor \\
&= \frac{29}{144}t^3 + \frac{13 + 5\cdot(-1)^{t-1}}{32}\cdot t + f + g
\end{aligned}$$

where

$$f = \begin{cases} 0 & \text{if } t \equiv 0 \mod 3 \\ \frac{1}{9} & \text{if } t \equiv 1 \mod 3 \\ -\frac{1}{9} & \text{if } t \equiv 2 \mod 3 \end{cases}$$

and

$$g = \begin{cases} 0 & \text{if } t \text{ is even} \\ \frac{1}{8}(-1)^{\frac{t-1}{2}} & \text{if } t \text{ is odd.} \end{cases}$$

□

Some results are

t	5	6	7	8	9
number of spanning trees in $K_{4,t}$	32,000	221,184	1,404,928	8,388,608	47,775,744
$I(K_{4,t})$	28	45	73	105	152

Chapter 5

Extreme cases

Looking at the number of non-isomorphic spanning trees, there are two extreme cases. First is the class of graphs that contain all non-isomorphic spanning trees. Of course, the complete graph does. But more interestingly: what is the smallest graph that contains them all?
The second extreme case is the graphs that contain just one non-ismorphic spannig tree. Also in this case, there are trivial graphs, like trees itself. But can we find bigger graphs?

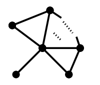
5.1 All non-isomorphic spanning trees

In this section, we look for graphs that contain all possible non-isomorphic spanning trees. In general this is not really interesting, since the complete graph always contains all possible non-isomorphic spanning trees. But, with n fixed, what is the *smallest* graph with that property?

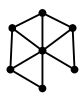
The first important thing is to determine all non-isomorphic (spanning) trees with n vertices. Of course, this can be done by applying algorithm NIST on K_n, but it can be done much faster. D.E. Knuth gives in [7] a way to determine all these trees.
With Algorithm O, he determines all rooted forests. This with use of sequences $[p_1, p_2, \ldots, p_n]$ ($p_i \in \{0, \ldots, n\}$) where p_i is the parent of vertex i. If $p_i = 0$, vertex i has no parent, so it is a root.
With exercise 90 of [7], we can expand this algorithm to determine all trees. First, we add the vertex 0 and connect all the roots with it. Then we get a

tree with $n+1$ vertices. Then we split the algorithms in two part: one will look for trees with one vertex as a center (which will be the vertex 0), and the other one for trees for which the center is an edge.

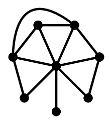
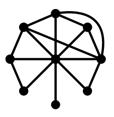
But if we know all non-isomorphic trees on n vertices, how do we get a smallest graph that contains them all? In general, two of the trees are always $K_{1,n-1}$ and P_{n-1}. So we start with combine those two in the "smallest way".



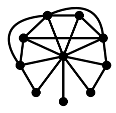
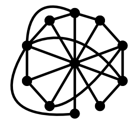
This is the smallest way, because $K_{1,n-1}$ contains only paths with two edges, so we have to add at least $n-2$ edges to get a path and in this case, we add $n-2$ edges.
For n is 3 to 6, this graph contains all non-isomorphic spanning trees. So for these n, we found the smallest graph.
Unfortunately, for $n=7$ we miss some spanning trees. But when we add one edge as below, we get a graph that contains all the non-isomorphic spanning trees. So also in this case we found a smallest graph.



For n is 8 and 9, we also found the smallest graph.

For n is 10 and 11, I found quite small graphs that contain all possible non-isomorphic spanning trees. But the algorithm was not fast enough to check whether the graphs are the smallest or not.

5.2 All spanning trees isomorphic

As said before, of course trees contain just one non-isomorphic spanning tree. But can we find bigger graphs with this property? And how many bigger graphs are there?

R. Fischer [3], L. Friess [4] and B. Zelinka [12] classified the graphs with exactly one isomorphism class of spanning trees. Before we can formulate that theorem, we have to define a special class of graphs.

Definition 5.1 *Let $A_1, A_2, \ldots, A_k$ be rooted trees. Then $C(A_1, A_2, \ldots, A_k)$ is the graph where the root of A_i is connected to the root of A_{i+1} by an edge. The indices of the trees are always* $\bmod k$*. So the root of A_k is connected to the root of A_1.*

So (if $k > 2$) in this graph we have one cycle, of length k, and every vertex of this cycle is the root of a tree.

Theorem 5.2 *Every graph with exactly one isomorphism class of spanning trees is either a tree or a graph of the form $C(A_1, A_2, \ldots, A_k)$, where A_i is a rooted tree and $A_i \cong A_{i+2}$ for $i = 1, 2, \ldots, k$.*

The first class (trees), we will call $\mathcal{T}$. The second class we will call $\mathcal{C}_k$ and $\mathcal{C}_k[n]$ are all graphs of $\mathcal{C}_k$ with exactly n vertices.

The most interesting class is of course $\mathcal{C}_k$ (or $\mathcal{C}_k[n]$). For which pairs (n, k) holds: $\mathcal{C}_k[n] \neq \emptyset$?

5.2.1 Conditions on k

We fix n. Since the k vertices have to be a cycle, we must have $k \geq 3$. Since n is the total number of vertices in the graph, also $k \leq n$ must hold.

5.2.1.1 Case: k odd

We have a graph $C(A_1, A_2, \ldots, A_k)$ with n vertices. As we have seen in theorem 5.2, we should have $A_i \cong A_{i+2}$ for $i = 1, 2, \ldots, k$.

Because k is odd, we can rewrite this condition into

$$A_i \cong A_{i+1} \text{ for } i = 1, 2, \ldots, k.$$

So all the A_i's have to be isomorphic.

The cycle contains k vertices, and since all A_i's has to be isomorphic, they all contain the same number of vertices: $\frac{n}{k}$. Since the number of vertices is in $\mathbb{N}$, $\frac{n}{k}$ has to be in $\mathbb{N}$. Hence, k has to be a divisor of n.

Theorem 5.3 *Let $n, k \in \mathbb{N}$ with k odd. Then*

$$\begin{aligned} k \nmid n &\Longrightarrow \mathcal{C}_k[n] = \emptyset \\ k|n &\Longrightarrow \mathcal{C}_k[n] = \{C(\underbrace{A, A, \ldots, A}_{k \text{ times}}) \mid A \text{ a rooted tree with } \tfrac{n}{k} \text{ vertices}\} \end{aligned}$$

Proof:
We showed above that k has to be a divisor of n, which proves $k \nmid n \Rightarrow \mathcal{C}_k[n] = \emptyset$.
If $k|n$, we have $\frac{n}{k} \in \mathbb{N}$. Let A be a rooted tree with $\frac{n}{k}$ vertices. Then $C(\underbrace{A, A, \ldots, A}_{k \text{ times}})$ satisfies "$A_i \cong A_{i+1}$ for $i = 1, 2, \ldots, k$", so the graph is in $\mathcal{C}_k[n]$. □

So "*k is a divisor of n*" is a sufficient condition for $\mathcal{C}_k[n]$ to be nonempty.

5.2.1.2 Case: k even

In this case we can rewrite the condition of theorem 5.2 into

$$A_1 \cong A_3 \cong A_5 \cong \cdots \cong A_{k-1}$$

and

$$A_2 \cong A_4 \cong A_6 \cong \cdots \cong A_k$$

Let m be the number of vertices of A_i with i even and l the number of vertices of A_i with i odd. Then

$$n = m\frac{k}{2} + l\frac{k}{2}.$$

So

$$\frac{2n}{k} = m + l$$

Because $m + l \in \mathbb{N}$, k has to be a divisor of $2n$.

Theorem 5.4 *Let $n, k \in \mathbb{N}$ with k even. Then*

$$k \nmid 2n \Longrightarrow \mathcal{C}_k[n] = \emptyset$$
$$k|2n \Longrightarrow \mathcal{C}_k[n] = \{C(\underbrace{A, B, A, B, \ldots, A, B}_{k \text{ trees}}) \text{ where } A, B \text{ are rooted trees with together } \tfrac{2n}{k} \text{ vertices}\}$$

Proof:
We showed above that k has to be a divisor of $2n$, which proves $k \nmid 2n \Rightarrow \mathcal{C}_k[n] = \emptyset$.
If $k|2n$, we have $\frac{2n}{k} \in \mathbb{N}$. Let A and B be rooted trees with together $\frac{2n}{k}$ vertices. Then $C(\underbrace{A, B, A, B, \ldots, A, B}_{k \text{ trees}})$ satisfies "$A_i \cong A_{i+2}$ for $i = 1, 2, \ldots, k$", so the graph is in $\mathcal{C}_k[n]$. $\square$

5.2.1.3 Graphs on n vertices

We define

$$\mathcal{C}[n] = \bigcup_{k=3}^{n} \mathcal{C}_k[n]$$

i.e. all graphs on n vertices that are not trees and with one isomorphism class of spanning trees. Then, with the results of section 5.2.1.1 and 5.2.1.2, we can conclude:

Theorem 5.5 *$\mathcal{C}[n]$ consists of all graphs*

1. *$C(\underbrace{A, A, \ldots, A}_{k \text{ times } A})$ where k is odd and a divisor of n, and A is a rooted tree with $\frac{n}{k}$ vertices.*

2. *$C(\underbrace{A, B, A, B, \ldots, A, B}_{k \text{ trees}})$ where k is even and a divisor of $2n$, and A and B are rooted trees with together $\frac{2n}{k}$ vertices.*

5.2.2 Number of graphs

If we fix n, we now know for which k there is a graph with one cycle (of length k) and one isomorphism class of spanning trees. But how many (not isomorphic) graphs on n vertices with one isomorphism class are there? Before we can determine that, we first determine how many graphs there are in $\mathcal{C}_k[n]$. Again, we have the two cases k *odd* and k *even*.

5.2.2.1 Case: k odd

As we have seen in section 5.2.1.1, the graphs we have to look at are: $C(\underbrace{A, A, \ldots, A}_{k \text{ times } A})$ where k is a divisor of n, and A is a rooted tree with $\frac{n}{k}$ vertices.

The number of graphs in this case is therefore the number of (non-isomorphic) rooted trees on $\frac{n}{k}$ vertices. This number is determined by D.E. Knuth in [6]: let a_m be the number of non-isomorphic rooted trees with m vertices. Obviously, $a_1 = 1$. If $m > 1$, the tree has a root and various subtrees. Suppose there are j_1 subtrees with 1 vertex, j_2 subtrees with 2 vertices, etc. Then we may choose j_k of the a_k possible k-vertex trees in $\binom{a_k+j_k-1}{j_k}$ ways. So

$$a_m = \sum_{j_1+2j_2+\cdots=m-1} \binom{a_1+j_1-1}{j_1} \cdots \binom{a_{m-1}+j_{m-1}-1}{j_{m-1}}$$

if $m > 1$. This number grows quite fast:

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a_m	1	1	2	4	9	20	48	115	286	719	1842	4766	12486	32973

5.2.2.2 Case: k even

As we have seen in section 5.2.1.2, the graphs we have to look at are: $C(\underbrace{A, B, A, B, \ldots, A, B}_{k \text{ trees}})$ where k is a divisor of $2n$, and A and B are rooted trees with together $\frac{2n}{k}$ vertices.

As in section 5.2.1.2, m is the number of vertices of A and l the number of vertices of B. Then, the number of graphs in this case is

$$a_{\frac{2n}{k}} + a_1 a_{\frac{2n}{k}-1} + a_2 a_{\frac{2n}{k}-2} + \cdots + a_{\lfloor \frac{2n}{k}/2 \rfloor} a_{\lceil \frac{2n}{k}/2 \rceil} = \sum_{m=0}^{\lfloor \frac{2n}{2k} \rfloor} a_m \cdot a_{\frac{2n}{k}-m}$$

(with $a_0 = 1$) since $l = \frac{2n}{k} - m$ and (for example) $m = 1, l = 0$ gives the same graph as $m = 0, l = 1$.

Example 5.6 *Take $n = 8$ and $k = 4$. Since $4|16 = 2 \cdot 8$, $\mathcal{C}_k[n]$ is nonempty. The number of graphs in $\mathcal{C}_k[n]$ is:*

$$\begin{aligned}\sum_{m=0}^{\lfloor \frac{2n}{2k} \rfloor} a_m \cdot a_{\frac{2n}{k}-m} &= \sum_{m=0}^{2} a_m \cdot a_{4-m} \\ &= a_0 \cdot a_4 + a_1 \cdot a_3 + a_2 \cdot a_2 \\ &= 1 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ &= 7\end{aligned}$$

5.2.2.3 Graphs on n vertices

Now we can combine the results of the sections 5.2.2.1 and 5.2.2.2.

Theorem 5.7

$$|\mathcal{C}[n]| = \sum_{\substack{k|n \\ k \text{ odd} \\ 3 \le k \le n}} a_{\frac{n}{k}} + \sum_{\substack{k|2n \\ k \text{ even} \\ 3 \le k \le n}} \sum_{m=0}^{\lfloor \frac{2n}{2k} \rfloor} a_m \cdot a_{\frac{2n}{k}-m}$$

We get

n	3	4	5	6	7	8	9	10	11	12	13	14	15
$\|\mathcal{C}[n]\|$	1	2	1	3	1	9	3	3	1	50	1	3	12

Example 5.8 *Take $n = 60$. The divisors of 60 are $1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60$. Since $k \geq 3$, we can forget the divisors 1 and 2.*
Then

$$\begin{aligned}
|\mathcal{C}[n]| &= \sum_{\substack{k|n \\ k \text{ odd} \\ 3\leq k\leq n}} a_{\frac{n}{k}} + \sum_{\substack{k|2n \\ k \text{ even} \\ 3\leq k\leq n}} \sum_{m=0}^{\lfloor \frac{2n}{2k} \rfloor} a_m \cdot a_{\frac{2n}{k}-m} \\
&= a_{20} + a_{12} + a_4 + \sum_{m=0}^{15} a_m \cdot a_{30-m} + \sum_{m=0}^{10} a_m \cdot a_{20-m} + \sum_{m=0}^{6} a_m \cdot a_{12-m} \\
&\quad + \sum_{m=0}^{5} a_m \cdot a_{10-m} + \sum_{m=0}^{3} a_m \cdot a_{6-m} + \sum_{m=0}^{2} a_m \cdot a_{4-m} + \sum_{m=0}^{1} a_m \cdot a_{2-m} \\
&= 12.826.228 + 4.766 + 4 + 697.878.253.158 + 25.088.231 + 9.191 \\
&\quad + 1.377 + 37 + 7 + 2 \\
&= 697.916.183.001
\end{aligned}$$

5.2.2.4 Special case: n is prime

For n is an odd prime, we see something special happens.
If k is odd, we know $k|n$ has to hold. But then $k = n$ (since $k \geq 3$). So the only graph in this case is the cycle with n vertices.
If k is even, then $k|2n$ has to hold. So, $k = 2$, $k = n$ or $k = 2n$. Since $3 \leq k \leq n$, $k = 2$ and $k = 2n$ can not hold and since k is even and n is odd (n is prime), also $k = n$ can not hold. So there is no graph for this case.

So

Theorem 5.9 *If n is an odd prime, $\mathcal{C}[n] = \{$cycle with n vertices$\}$*

Chapter 6

References

1. G.E. Andrews, *The theory of partitions*, 1976
page 81

2. D.I.A. Cohen, *Basic techniquens of combinatorial theory*, John Wiley & Sons, 1978
page 116

3. R. Fischer, *Über Graphen mit isomorphen Gerüsten*, Monatshefte für Mathematik 77 (1973)
pages 24-30

4. L. Friess, *Graphen, worin je zwei Gerüste isomorph sind*, Math. Ann. 204 (1973)
pages 65-71

5. C. Jordan, *Sur les assemblages de lignes*, J. reine und angew. Math. 70 (1869)
pages 185-190

6. D.E. Knuth, *The art of computer programming, volume 1*, 3rd edition, Addison-Wesley (1997)
pages 386-399

7. D.E.Knuth, *The art of computer programming, volume 4* fascicle 4, section 7.2.1.6

8. J. Köbler, U. Schöning and J. Torán, *The graph isomorphism problem: its structural complexity*, 1993
pages 5-6

9. A. Mohr, *Partitioning the labeled spanning trees of an arbitrary graph into isomorphism classes*, Master thesis, 2008

10. J.W. Moon, *Counting labelled trees*, 1970
Chapter 5: *The matrix tree theorem* (pages 39-51).

11. H. Prüfer, *Neuer Beweis eines Satzes über Permutationen*, Arch. Math. Phys. 27
pages 742-744

12. B. Zelinka, *The graphs, all of whose spanning trees are isomorphic to each other*, Casopis Pest. Mat. 96, (1971)
pages 33-40

13. http://www.math.ru.nl/∼bosma/Students/JannekevandenBoomen/